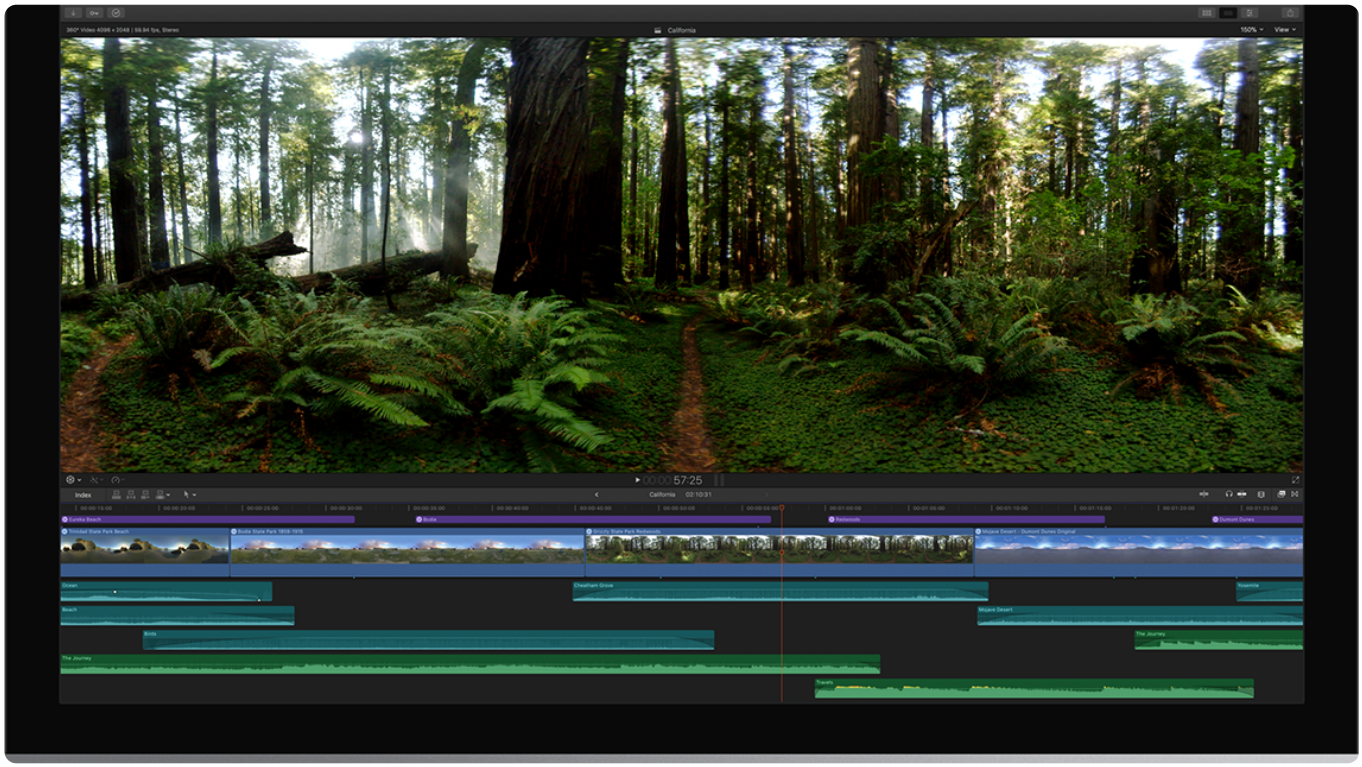
04｜编码原理：视频究竟是怎么编码压缩的？

2021-11-29 李江

《攻克视频技术》

课程介绍 >



讲述：李江

时长 20:28　大小 18.75M

你好，我是李江。今天我们一起来聊一聊视频编码。

说到视频，我们首先想到的可能就是占内存。我们知道一个视频是由一连串图像序列组成的，视频中图像一般是 YUV 格式。现在，我们假设有一个电影视频，分辨率是 1080P，帧率是 25fps，并且时长是 2 小时，如果不做视频压缩的话，它的大小是 1920 x 1080 x 1.5 x 25 x 2 x 3600 = 260.7G。而我们的电脑一般是 500G 的硬盘，那就连 2 部电影都放不下了。如果是在视频通话场景下的话，按照这个大小去传输视频数据，对流量和带宽资源的消耗也是非常大的，并且如此大的数据发送对网络要求也非常高，很显然我们是接受不了的。因此，做视频编码压缩就非常有必要。

那么，接下来我就深入讲讲视频编码，带你从预测编码、变换编码、熵编码等方面，系统了解下视频编码的原理。相信这节课过后，你会对视频编码有一个全面的了解。

视频编码的原理

视频编码是对一帧帧图像来进行的。一般我们所熟知的彩色图像的格式是 RGB 的，即用红绿蓝三个分量的组合来表示所有颜色。但是，RGB 三个颜色是有相关性的，为了去掉这个相关性，减少需要编码的信息量，我们通常会把 RGB 转换成 YUV，也就是 **1 个亮度分量和 2 个色度分量**。

另外，人眼对于亮度信息更加敏感，而对于色度信息稍弱，所以视频编码是将 Y 分量和 UV 分量分开来编码的。

而对于每一帧图像，又是划分成一个个块来进行编码的，**这一个个块在 H264 中叫做宏块**，而在 VP9、AV1 中称之为超级块，其实概念是一样的。宏块大小一般是 16x16（H264、VP8），32x32（H265、VP9），64x64（H265、VP9、AV1），128x128（AV1）这几种。这里提到的 H264、H265、VP8、VP9 和 AV1 都是市面上常见的编码标准，下面我会介绍，这里就不再详细讲述。

图像一般都是有数据冗余的，主要包括以下 4 种：

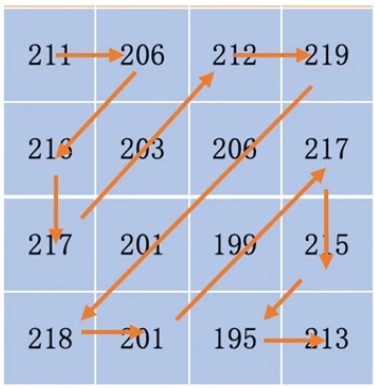
空间冗余。比如说将一帧图像划分成一个个 16x16 的块之后，相邻的块很多时候都有比较明显的相似性，这种就叫空间冗余。

时间冗余。一个帧率为 25fps 的视频中前后两帧图像相差只有 40ms，两张图像的变化是比较小的，相似性很高，这种叫做时间冗余。

视觉冗余。我们的眼睛是有视觉灵敏度这个东西的。人的眼睛对于图像中高频信息的敏感度是小于低频信息的。有的时候去除图像中的一些高频信息，人眼看起来跟不去除高频信息差别不大，这种叫做视觉冗余。

信息熵冗余。我们一般会使用 Zip 等压缩工具去压缩文件，将文件大小减小，这个对于图像来说也是可以做的，这种冗余叫做信息熵冗余。
视频编码就是通过减少上述 4 种冗余来达到压缩视频的目的。接下来我们就一起来慢慢剥开视频编码这个“洋葱”吧。

对于一个 YUV 图像，我们把它划分成一个个 16x16 的宏块（以 H264 为例），Y、U、V 分量的大小分别是 16x16、8x8、8x8。这里我们只对 Y 分量进行分析（U、V 分量同理）。假设 Y 分量这 16x16 个像素就是一个个数字，我们从左上角开始之字形扫描每一个像素值，则可以得到一个“像素串”。如下图所示：



之字形扫描方式

极客时间

如果你是程序员的话，你肯定做过一个压缩字符串的编程题目，就是将“aaaabbbccccc” 压缩成 “4a3b5c”，字符串由 13 个字节压缩到 7 个字节，这个叫做**行程编码**。如果我们对图像宏块扫描出来的这个“像素串”做同样的行程编码操作，是不是也有可能减小图像块呢？

在对“像素串”行程编码之前，我们先回过头来看看另一个行程编码的例子。如果刚才编程题的字符串是 “abcdabcdabcd” 的话，那么编码之后就会是“1a1b1c1d1a1b1c1d1a1b1c1d”。字符串的大小从 13 字节变成了 25 字节，还变大了。

我们发现如果想要达到压缩的目的，我们必须要使得编码前的字符串中出现比较多连续相同的字符。这对于图像块也是一样的。我们需要使得扫描出来的“像素串”，也尽量出现连续相同的像素值，最好是一连串数字很小（比如 0）的“像素串”，因为 0 在二进制中只占 1 个位就可以了。

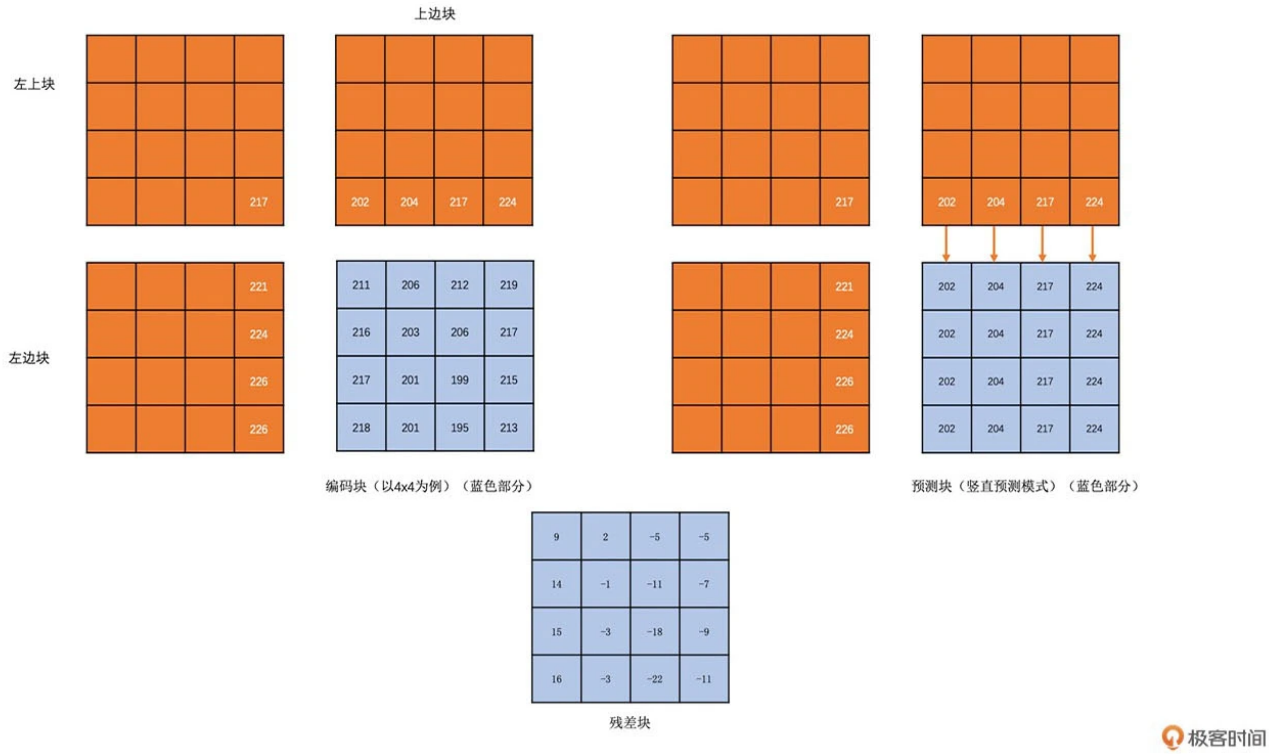
这个地方你可能会有疑惑，0 也是至少要一个字节存储，需要 8 个位，怎么会是 1 个位呢？这个有的编码算法是可以做到的，比如指数哥伦布编码，它就可以做到 0 只占用一个位。事实上，算术编码可以做到一个符号只占用 0 点几个位，连一个位都不用，这里不详细展开了，感兴趣的话你可以去查阅下资料。

那我们**如何做到将这串像素值变成有很多 0 的“像素串”呢**？

首先第一步，**我们通过减少图像块的空间冗余和时间冗余来接近这个目标**。刚才我们也说到，图像内部相邻宏块之间有很多相似性，并且两张图像之间也有很多相似性。因此，根据图像的这个特点，我们可以在编码的时候进行**帧内预测和帧间预测**。

帧内预测就是在当前编码图像内部已经编码完成的块中找到与将要编码的块相邻的块。一般就是即将编码块的左边块、上边块、左上角块和右上角块，通过将这些块与编码块相邻的像素经过多种不同的算法得到多个不同的预测块。

然后我们再用编码块减去每一个预测块得到一个个残差块。最后，我们取这些算法得到的残差块中像素的绝对值加起来最小的块为预测块。而得到这个预测块的算法为帧内预测模式。



由于这个残差块中像素的绝对值之和最小，这个残差块的像素值经过扫描之后的“像素串”是不是就比直接扫描编码块的“像素串”中的像素值更接近 0 了？

同理，帧间预测也是一样的。我们在前面已经编码完成的图像中，循环遍历每一个块，将它作为预测块，用当前的编码块与这个块做差值，得到残差块，取残差块中像素值的绝对值加起来最小的块为预测块，预测块所在的已经编码的图像称为参考帧。预测块在参考帧中的坐标值 (x0, y0) 与编码块在编码帧中的坐标值 (x1, y1) 的差值 (x0 - x1, y0 - y1) 称之为运动矢量。而在参考帧中去寻找预测块的过程称之为运动搜索。事实上编码过程中真正

的运动搜索不是一个个块去遍历寻找的，而是有快速的运动搜索算法的。之后我们在帧间预测的课中会详细介绍。

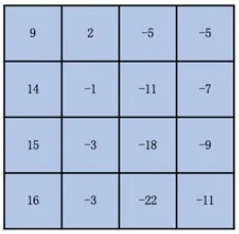
总之，通过预测得到的残差块的像素值相比编码块的像素值，去除了大部分空间冗余信息和时间冗余信息，这样得到的像素值更小。如果把这个残差块做扫描得到的像素串送去做行程编码，是不是相比直接拿编码块的像素串去做编码更有可能得到更大的压缩率？

但是我们的目标不只是将像素值变小，而是希望能出现连续的 0 像素，那怎么办呢？

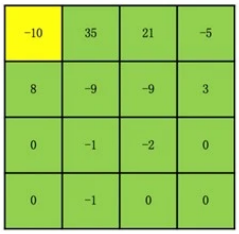
这就需要利用我们人眼的视觉敏感性的特点了。我们刚才说了人眼对高频信息不太敏感。因为人眼看到的效果可能差别不大，所以我们可以去除一些高频信息。这个就是接下来我们要讨论的 **DCT 变换和量化**。

为了分离图像块的高频和低频信息，我们需要将图像块变换到频域。常用的变换是 DCT 变换。DCT 变换又叫离散余弦变换。在 H264 里面，如果一个块大小是 16x16 的，我们一般会划分成 16 个 4x4 的块（当然也有划分成 8x8 做变换的，我们这里以 4x4 为例）。然后对每个 4x4 的块做 DCT 变换得到相应的 4x4 的变换块。

变换块的每一个“像素值”我们称为系数。变换块左上角的系数值就是图像的低频信息，其余的就是图像的高频信息，并且高频信息占大部分。低频信息表示的是一张图的总体样貌。一般低频系数的值也比较大。而高频信息主要表示的是图像中人物或物体的轮廓边缘等变化剧烈的地方。高频系数的数量多，但高频系数的值一般比较小（注意不是所有的高频系数都一定小于低频，只是大多数高频系数比较小）。如下图所示（黄色为低频，绿色为高频）：



残差块



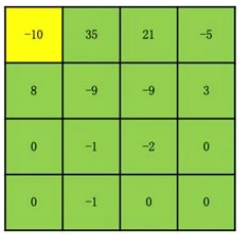
DCT变换后的块

极客时间

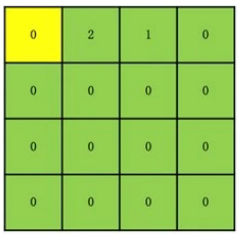
这样做完了 DCT 变换之后，低频和高频信息就分离开来了。由于低频信息在左上角，其余的都是高频信息。那么如果我们对变换块的像素值进行“之字形”扫描，这样得到的像素串，前面的就是数值比较大的低频系数，后面就是数值比较小的高频部分。

由于人眼对高频信息不太敏感，如果我们通过一种手段去除掉大部分高频信息，也就是将大部分高频信息置为 0，但又不太影响人的观感，是不是就可以达到我们最初的目标，即可以得到有一连串 0 的像素串？这就涉及到量化操作了。

我们让变换块的系数都同时除以一个值，这个值我们称之为**量化步长**，也就是 QStep（QStep 是编码器内部的概念，用户一般使用**量化参数** QP 这个值，QP 和 QStep 一一对应，你可以自行去网上查询一下转换表），得到的结果就是量化后的系数。**QStep 越大，得到量化后的系数就会越小**。同时，相同的 QStep 值，高频系数值相比低频系数值更小，量化后就更容易变成 0。这样一来，我们就可以将大部分高频系数变成 0。如下图所示：



DCT变换后的块



QP取26（QStep为13）量化后（使用H264标准）

极客时间

解码的时候，我们会将 QStep 乘以量化后的系数得到变换系数，很明显这个**变换系数和原始没有量化的变换系数是不一样的**，这个就是我们常说的有损编码。而到底损失多少呢？

这由 QStep 来控制，QStep 越大，损失就越大。QStep 跟 QP 一一对应，也就是说确定了一个 QP 值，就确定了一个 QStep。所以从编码器应用角度来看，**QP 值越大，损失就越大，从而画面的清晰度就会越低**。同时，QP 值越大系数被量化成 0 的概率就越大，这样编码之后码流大小就会越小，压缩就会越高。

以上就是视频编码的推理过程。总结一下就是，为了能够在最后熵编码的时候压缩率更高，我们希望送到熵编码（以行程编码为例）的“像素串”，是一串含有很多 0，并且最好连续为 0 的“像素串”。

为了达到这个目标，我们先通过帧内预测或者帧间预测去除空间冗余和时间冗余，从而得到一个像素值相比编码块小很多的残差块。之后我们再通过 DCT 变换将低频和高频信息分离开来得到变换块，然后再对变换块的系数做量化。由于高频系数通常比较小，很容易量化为 0，同时人眼对高频信息不太敏感，这样我们就得到了一串含有很多个 0，大多数情况下是一串含有连续 0 的“像素串”，并且人的观感还不会太明显。这样，最后熵编码就能把图像压缩成比较小的数据，以此达到视频压缩的目的。这就是**视频编码的原理**。

编码器的对比及选择

说完了编码器的原理，那么常用的编码标准有哪些呢？它们的区别又是什么？我们怎么选择合适的编码器？

现在市面上常见的编码标准有 H264、H265、VP8、VP9 和 AV1。目前 H264 和 VP8 是最常用的编码标准，且两者的标准非常相似。H265 和 VP9 分别是他们的下一代编码标准，这两个标准也非常相似。AV1 是 VP9 的下一代编码标准。

H264 和 H265 是需要专利费的，而 VP8 和 VP9 是完全免费的。由于 H265 需要付高额的版权费，以谷歌为首的互联网和芯片巨头公司组织了 AOM 联盟，开发了新一代压缩编码算法 AV1，并宣布完全免费，以此来对抗高额专利费的 H265。

目前普通产品还是使用 H264 最多，而 H265 因为专利费使用得比较少。VP8 是 WebRTC 默认的编码标准，且 WebRTC 使用 VP8 最多。同时，WebRTC 也支持 VP9 和 AV1。YouTube 使用了 VP9 和 AV1。Netflix 也使用了 AV1。我们下面来比较一下 H264、H265、AV1 这三代编码算法的区别。现在你不理解没有关系，等你学完后面的课程会很容易明白下面表格中所涉及的知识的。

编码标准	块大小和划分方式	帧内编码	帧间编码	变换	熵编码	滤波和后处理
H264	最大16x16 可划分成8X16、16X8、8X8、4X8、8X4、4X4	8个方向模式 +planar+DC模式	中值MVP	DCT 4X4/8X8	CAVLC、CABAC	去块滤波
H265	最大支持64x64， 四叉树划分	33个方向模式 +planar+DC模式	Merge模式 AMVP模式	DCT 4X4/8X8/16X16/32X32 DST 4X4	CABAC	去块滤波 SAO滤波
AV1	最大支持128x128， 四叉树划分	56个方向模式+3个平滑模式+递归FilterIntra模式+色度CFL模式+调色板模式+帧内块拷贝模式	OBMC+扭曲运动补偿 +高级复合预测+复合帧内预测	4x4-64x64正方形 +1:2/2:1+1:4/4:1矩形 DCT/ADST/flipADST/IDTX	多符号算术编码	去块滤波 CDEF LR滤波 Frame超分 Film Grain

极客时间

从上面表格中可以看到，标准越新，最大编码块就越大，块划分的方式也越多，编码模式也就越多。因此压缩效率也会越高，但是带来的编码耗时也越大。所以在选择编码器的时候需要根据自己的实际应用场景来选择，同时还需要考虑专利费的问题。还有一个就是考虑有没有硬件支持的问题。目前 H264 和 H265 的硬件支持已经很好了，AV1 才刚开始，硬件支持较少，之后会有更多硬编硬件支持。

我做了个简单的编码清晰度和耗时对比，都是在软件编码下进行的。具体如下表所示。我们可以看到相同码率下，AV1 清晰度稍好于 H265，而 H264 最差，但是编码耗时则相反，AV1 最高，H265 次之，H264 速度最快。

编码器	H264	H265	AV1
PSNR	29.3	31.2	32
速度	25fps	8fps	3fps

PSNR是峰值信噪比，用来客观地表示图片的质量

极客时间

所以，如果是在性能比较差的机器上编码，最好使用 H264 和 VP8 等速度快的编码器。如果是在比较新的机器上，可以考虑 H265 编码。中等机器如果支持 H265 硬编也是可以考虑的。但有一个问题就是 H265 需要考虑专利费的问题，同时浏览器原生不支持 H265 编码，所以有这方面需求的，最好不要使用 H265，可以考虑使用 VP9，甚至可以考虑 AV1。另外，由于 AV1 原生标准就支持屏幕编码的优化，所以屏幕编码场景下可以考虑使用 AV1 编码。

小结

总结一下，我们今天主要讲了视频编码的必要性，以及视频编码的原理。**视频编码主要分为熵编码、预测、DCT 变换和量化这几个步骤。**

1. 熵编码（以行程编码为例）：视频编码中真正实现“压缩”的步骤，主要去除信息熵冗余。在出现连续多个 0 像素的时候压缩率会更高。
2. 帧内预测：为了提高熵编码的压缩率，先将当前编码块的相邻块像素经过帧内预测算法得到帧内预测块，再用当前编码块减去帧内预测块得到残差块，从而去掉空间冗余。
3. 帧间预测：类似于帧内预测，在已经编码完成的帧中，先通过运动搜索得到帧间预测块，再与编码块相减得到残差块，从而去除时间冗余。
4. DCT 变换和量化：将残差块变换到频域，分离高频和低频信息。由于高频信息数量多但大小相对较小，又人眼对高频信息相对不敏感，我们利用这个特点，使用 QStep 对 DCT 系数进行量化，将大部分高频信息量化为 0，达到去除视觉冗余的目的。

这里你需要注意的是，视频编码实际的步骤是预测、DCT 变换和量化，最后是熵编码。经过这几步操作之后，视频中的冗余信息大部分被去除，达到了编码压缩的效果。当然，如

何做帧内预测和帧间预测？如何找到最优的预测块？DCT 变换和量化又是怎么做的呢？敬请期待我们接下来的课程，我会和你细聊。

思考题

现在请你思考一下，视频编码过程中，一帧图像能同时进行帧内预测和帧间预测吗？

你可以把你的答案和感受写下来，分享到留言区，与我一起讨论。下节课再见。

分享给需要的人，Ta订阅后你可得 20 元现金奖励

生成海报并分享

赞 1 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 03 | 缩放算法：如何高质量地缩放图像？

下一篇 05 | 码流结构：原来你是这样的H264

精选留言 (6)

写留言

ForwardsHao
2021-11-29

一帧图像即存在空间冗余又存在时间冗余，所以是帧间预测和帧内预测都是可以同时可以在一帧上应用的。这样一个编码的宏块，都会即由本帧内的前面的宏块又由相关帧的预测块影响。

> 作者回复: 是的，这个思路很正确。但是I帧是不能进行帧间预测的。因为帧间预测是需要依赖于参考帧的，这样肯定需要一开始有一个帧是可以独立的编解码的。不然大家都相互依赖了。

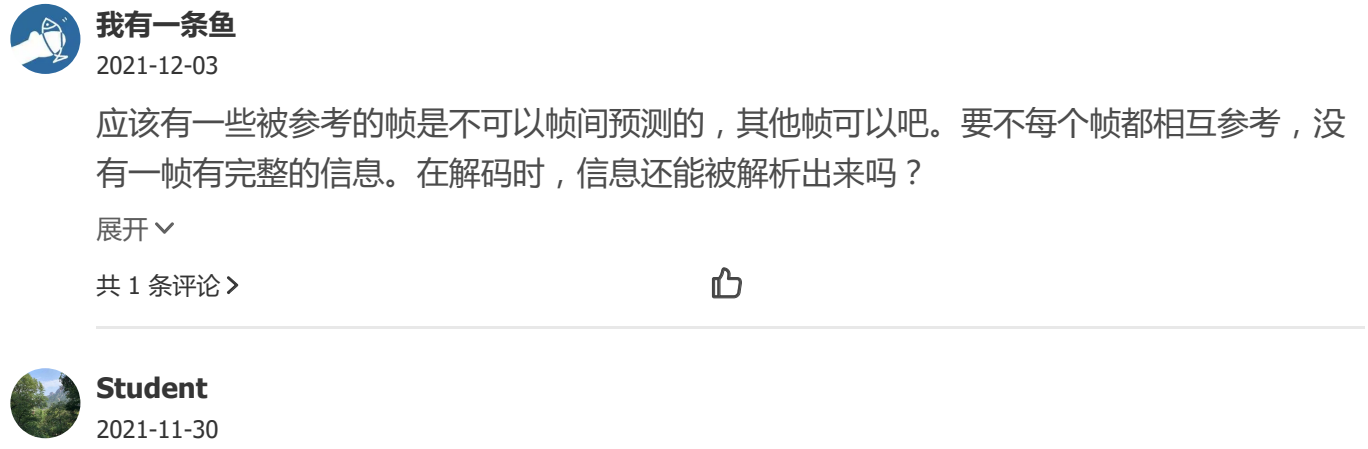
1

我有一条鱼
2021-12-03

上面中写av1和H265的编码速度是3fps和8fps。这样的速度是怎么在直播场景中使用的吗？在页面上展示的时候不是15fps左右吗？

展开

作者回复: 这个怪我没有解释清楚。这个对比看相对数据就可以。因为我选用的速度档是很慢速的。编码器是可以选择编码速度的。比如说x264和x265都有一个preset参数可以设置编码速度的。一般实际不会使用这种速度的。真实场景的编码速度需要看编码器设置，编码设备和编码画面的复杂度等多个方面来决定的。



我有一条鱼

2021-12-03

应该有一些被参考的帧是不可以帧间预测的，其他帧可以吧。要不每个帧都相互参考，没有一帧有完整的信息。在解码时，信息还能被解析出来吗？

展开

共 1 条评论 >

Student

2021-11-30

1920 x 1080 x 1.5 x 25 x 2 x 3600 = 260.7G。其中的1.5是怎么来的 谢谢

作者回复: 1.5是一般我们图像常用的是YUV420格式的，YUV420格式每一个像素对应了1个字节的Y、0.25个字节的U和0.25个字节的V。所以一个像素占用1.5个字节。你可以学习一下第二节课里面有详细的内容来讲述。

共 2 条评论 >

被讨厌的勇气

2021-11-29

应该可以。帧内预测和帧间预测都是以块为基本单元的，而一帧包含多个块，所以，可以将帧间预测与帧内预测施加到同一帧的不同块上

展开

作者回复: 这个"以块为基本单元"的思考这个问题的方式很好，很棒。因为确实是分块来选择模式的。但是I帧不能用帧间预测。P帧和B帧两种预测方式都可以。

西钾钾

2021-11-29

应该可以，比如I帧。

展开

作者回复: I帧只能进行帧内预测，因为I帧需要能够自己独立编解码，如果使用帧间预测就有依赖了。P帧既可以帧间预测又可以帧内预测。

共 2 条评论